FSE PROGRAM SUMMARY DOCUMENT #4

Transfer Function Routine XFRSET

R. E. McFarland

October 1990

NASA

National Aeronautics and
Space Administration

Document: FSE PROGRAM SUMMARY #4, orig. Feb. 14, 1990, final Oct. 29, 1990

Title: Transfer Function Routine XFRSET

Author: R. E. McFarland, NASA

Function: Computation of the state transition matrix and forcing function coefficient vector for single-input, single-output transfer functions.

Availability: (1) SIMDEV VAX, $DISK1:[STRIKE]XFRSET.FOR
(2) FSD VAX, $LIBRARY:[STRIKE]XFRSET.FOR
(3) ADDEV VAX, $DISK1:[MCFARLAND.STRIKE]XFRSET.FOR
(4) Author, PC Diskette

Language: FORTRAN, Computer Portable, ANSI Standard

References: (1) State Variables for Engineers, Paul M. DeRusso, Rob J. Roy and Charles M. Close, Wiley & Sons, Inc., New York, 1967.

(2) On Optimizing Computations for Transition Matrices, R. E. McFarland and A. B. Rochkind, IEEE Trans. Automatic Control, Vol. AC-23, No. 3, June 1978.

Summary

This document outlines software for the discrete *state space solution* of single-input, single-output transfer function given by,

$$\frac{x(s)}{u(s)} = \frac{1}{s^N + a_N s^{N-1} + \ldots + a_2 s + a_1}$$

The state space solution of this equation requires the computation of both the *transition matrix* and *forcing function vector*. Once available, the more general problem of ratios of polynomial transfer functions may be solved. This is permitted because the state space technique develops all of the "N" individual states of the system.

The state space solution to transfer functions is invariably better than sequential integration techniques. For example, the state space solution to a stable transfer function is always stable, whereas the same cannot be said for other techniques.

This document is the first of three in a series. XFRSET, the program discussed in this document, produces the transition matrix and forcing function vector. If the denominator coefficients a_n are constants, this program need be called only once per simulation. In the case of nonstationary coefficients, the state space solution technique itself only makes sense if the coefficients are *slowly varying*, such as in the case of functions of altitude or velocity. In this case XFRSET is only called periodically, not every cycle.

The second document in this series concerns the program XFRRUN. It involves considerations of numerator coefficients and "data holds." XFRRUN is used for the transition of states in real time simulations, and is necessarily called every computer cycle. This transition requires, of course, the transition matrix and forcing function vector.

The third document in this series concerns the program XFRBOD. This program is not intended for real time use. It produces Bode plot information, and should be quite useful in comparing the characteristics of discrete realizations to original continuum transfer functions.

State Space Form

The cited transfer function arises from the ordinary N^{th} order differential equation,

$$u(t) = \frac{d^N x(t)}{dt^N} + a_N \frac{d^{N-1}x(t)}{dt^{N-1}} + a_{N-1} \frac{d^{N-2}x(t)}{dt^{N-2}} + \ldots + a_2 \frac{dx(t)}{dt} + a_1 x(t)$$

The states for the N^{th} order system are defined here as the outputs of successive integrations. This permits the equation to be written,

$$\frac{dx_N}{dt} = u(t) - a_N x_N - a_{N-1} x_{N-1} - \ldots - a_2 x_2 - a_1 x_1$$

where all but the N^{th} derivative are contained within the state definition,

$$x_n = \frac{d^{n-1}x(t)}{dt^{n-1}} \qquad (n = 1, 2, \ldots, N)$$

The differential equation then has a convenient matrix form[1],

$$\dot{x}(t) = Ax(t) + U(t)$$

with an implied companion matrix equation (introduced in FSE Program Summary #5),

$$y(t) = Bx(t)$$

that will permit the solution to the more general ratio of polynomial Laplace equations:

$$\frac{y(s)}{u(s)} = \frac{b_{M+1}s^M + b_M s^{M-1} + \ldots + b_2 s + b_1}{s^N + a_N s^{N-1} + \ldots + a_2 s + a_1}$$

The numerator polynomial is not discussed here. However, it should be noticed that all of the states are required for the solution to this more general function.

In the matrix form of the differential equation the matrix A is the essential matrix of the system expressed in canonical form,

$$[A] = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_1 & -a_2 & -a_3 & \cdots & -a_N \end{bmatrix}$$

and the input is just a scalar,

$$U(t) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ u(t) \end{bmatrix}$$

Calling Sequence

```
CALL XFRSET(T,N,A,NTERM,SETB)
```

The calling sequence contains the following quantities:

T	The (input) cycle time, in seconds.
N	The (input) order of the denominator s-plane polynomial ($1 \leq N \leq 20$ is the current software dimension limitation).
A( )	The (input) vector dimensioned N (or greater), consisting of the denominator coefficients a_n. A(N+1) is not included; it is always assumed unity.
NTERM	An XFRSET scalar output, available to the user for an examination of series convergence properties. (See Reference 2).
SETB( )	A buffer unique to the denominator of dimension of at least $N^2 + N + 1$. The first N+1 cells of this buffer contain the forcing function coefficients and the remaining N^2 cells contain the transition matrix. This output buffer, explained below, is computed by XFRSET. It is a function of both the cycle time T and the coefficients a_n.

The subroutine XFRSET concerns itself only with the denominator polynomial. The cycle time (T), the order of the denominator polynomial (N), and the specified denominator coefficients a_n (n = 1, 2, ... N) are used to create the buffer SETB, which must be dimensioned at least $N^2 + N + 1$ in the calling program. Note that it generally takes N

+ 1 coefficients to express an N^{th} order polynomial - but because "N" is specified the leading coefficient a_{N+1} is always assumed unity, and is not included in the A( ) dimension requirements.

XFRSET is called initially, and for nonstationary transfer functions may be called periodically, or whenever the denominator coefficients change "significantly." Although "significantly" is not a well defined term in this context, the slowly-varying coefficient hypothesis would have it that coefficient frequency content be much lower than the input/output frequency content. Also, engineering judgement dictates that discrete changes in the location of the poles should not excite the transfer function. Hence, for nonstationary transfer functions some care should be exercised in the selection of the period of calling XFRSET.

If the call to XFRSET must be made every cycle, then it is doubtful that the transfer function is itself amenable to a Laplace representation. Also, calls to XFRSET are computationally expensive relative to the required XFRRUN calls. The penalty for redundant XFRSET calls is computational workload.

The Buffer SETB

As created by a call to XFRSET, the buffer SETB contains the forcing function coefficients in a vector [Λ],

$$[\Lambda] = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \\ \lambda_{N+1} \end{bmatrix}$$

which are contained in the first N + 1 cells of SETB. The transition matrix [Γ] comprises the remaining N^2 cells of SETB (for a total of $N^2 + N + 1$):

$$[\Gamma] = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \cdots & \gamma_{1N} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & \cdots & \gamma_{2N} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & \cdots & \gamma_{3N} \\ \vdots & \vdots & \vdots & & \vdots \\ \gamma_{N1} & \gamma_{N2} & \gamma_{N3} & \cdots & \gamma_{NN} \end{bmatrix}$$

The above vector and matrix are sufficient for the complete solution to the state variable equations, regardless of the selected data hold. Hence, SETB contains the generalized coefficients for the discrete realization of a linear transfer function. These coefficients are computed to the accuracy of the computer word length.

XFRSET's buffer SETB is required by subroutine XFRRUN, which performs the actual transition of the states during real-time simulation. This buffer is also required by subroutine XFRBOD in a non-real time analysis of the performance characteristics

of the discrete realization. As a preview of the discrete operations, three different data-hold formulations are presented below. This preview is provided to show the general nature of XFRSET outputs; they may be used for data-hold formulations beyond those provided by XFRRUN.

Transition

"Transition" consists of the discrete operation of determining the numerical output(s) of a transfer function, given the input. This output may be computed as applicable to the next time point (advancing form), or applicable at the same time as the input (concurrent form). This determination is made when a "data hold" is selected, *i. e.*, the characteristics of the input data *between the sampling instants* is specified. The state transition technique requires that a data hold assumption be made. Three separate data holds are handled by XFRRUN - but this does not preclude the possibility of additional data holds.

The three data holds are presented in Fig. 1. This figure shows the consequences of sampling only ten times per cycle of input data, for each of the data holds.

Zero-Order Hold

If the zero-order data hold is selected, the input data is assumed to be constant *during the (new) interval*, so that the output point of applicability is the end of the interval (advance). If, for some reason, the output point were required to be concurrent with the input point, and the zero-order hold formulation was absolutely required, then the input itself would have to be delayed by one cycle. In practice, the zero order hold has application for random inputs, such as in turbulence models, and in cases where the output advance is required, but the first order data hold is numerically unstable.

For step inputs the zero-order data hold provides exact answers. In this unusual case the data hold assumption just happens to perfectly describe the input functionality. However, this rarely occurs in practice. Hence, the utility of the zero-order data hold has become exaggerated. In fact, for much more general inputs that are representative samples of continuum behavior, the zero-order hold formulation actually advances the output by half of a cycle, rather than a full cycle. This can lead to some unexpected results unless care is taken.

First-Order Hold

If the first-order data hold is selected, the input data is assumed to have straight-line behavior. The current and previous input data values are used to project this linear behavior during the new interval, in a process usually called "extrapolation." This then produces an output that is applicable at the end of the interval (advance), similar to that of the zero-order data hold. In real-time simulation work an advanced output is required only in certain internal feedback paths (with nonlinearities) and occasionally in algebraic loops. Most notably, however, the "first integration" in the kinematic module STRIKE (or SMART) uses the first-order data hold. This process results in the Adams-Bashforth algorithm, which advances the vehicle translational and angular velocities. This very fact generally relieves Ames programmers from having to consider "advancing forms" when they code the individual modules in an aircraft simulation. The outputs (all forces and moments) delivered to the kinematic module should be *concurrent*

$$H_0(s) = \frac{1 - e^{-sT}}{s}$$

$$H_1(s) = \left(\frac{1 - e^{-sT}}{s}\right)^2 \left(\frac{1+sT}{T}\right)$$

$$H_2(s) = \left(\frac{1 - e^{-sT}}{s}\right)^2 \frac{e^{sT}}{T}$$

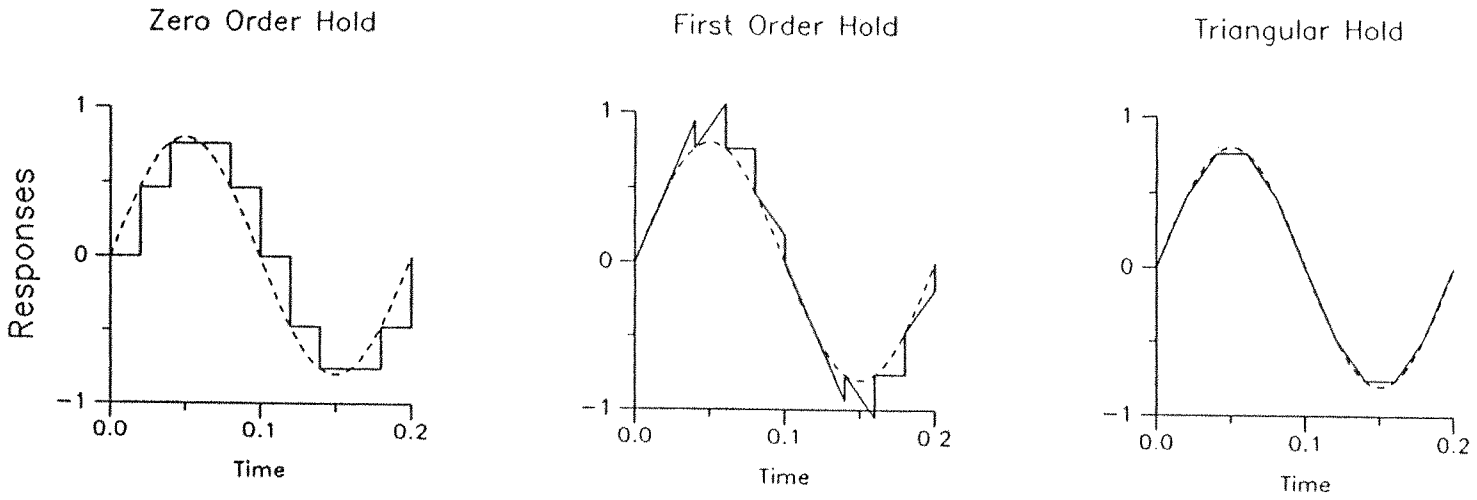


Fig. 1 – DATA HOLDS

with the input point (pilot input A/D), and this is not possible using a first-order data hold, unless the input data values are intentionally delayed by one cycle. A silly operation.

Triangular Hold

If the triangular data hold is selected, the input data is also assumed to be a straight line during the interval, but the input data is not extrapolated, it is interpolated. This produces a significant difference from the first-order data hold. The output is not applicable at the end of the interval, it is applicable at the beginning of the interval, *i. e.*, the same time point as the most recent input (concurrent). The triangular data hold is thus the algorithm of choice. If it is not used, then some compelling reason must exist.

If an advance is required, the first-order data hold usually performs the job better than the zero-order data hold. However, it amplifies the higher frequency components more, and this may adversely affect the gain margin in certain feedback paths. If an advance is not required, which is the usual case when internal feedback paths are not of concern, then the gain and phase characteristics of the triangular hold are *wonderful.*

The Bilinear transform technique, sometimes called "Tustin," is not a state space solution technique (it is an algebraic substitution technique), and is therefore not discussed here. Although it is a "concurrent form" similar to the triangular hold, it delivers inferior performance because of frequency warping. Whenever possible, it should be avoided.

Transition thus depends upon the selected data hold (here called IHOLD), which is the selected assumption concerning the behavior of the input data *between the sampling points.* The difference between an "advancing" data hold and a "concurrent" data hold should be well understood.

Transition is not performed by XFRSET, it is performed by XFRRUN. Alternately, it may be performed by the user's own code. The transition process is here previewed in order to give insight into the procedure, and to show the importance of the transition matrix and forcing function vector.

For simulation models at Ames, the *outputs* of any aero, control or propulsion module should be concurrent with the input point, or beginning of the cycle. This is because these outputs are generally proportional to forces or moments, which are accumulated by the kinematic module and assumed to be coincident with the pilot input. At the end of the cycle the kinematic module combines these individual module outputs, applicable at t_k, and integrates them to velocities and positions, applicable at t_{k+1}, or the beginning of the next interval (or end of the current interval). Hence, *advances in module outputs, wherein transfer functions usually reside, are erroneous.* Gross errors in the time subscript, sometimes called "temporal index" problems, produce phase errors up to 180°.

Although advances in *total* module outputs are thus almost invariably undesirable, this does not mean that advancing options are not useful. Indeed, *internal* to various modules they are often required, especially in feedback paths. As will be shown in FSE Summary Document #5, this sometimes requires dual XFRRUN calls, but never requires additional XFRSET calls.

The three basic data hold options used at Ames are (1) IHOLD = 0, the zero-order data hold, which advances the output by one cycle, (2) IHOLD = 1, the first-order data hold, which also advances the output by one cycle, and (3) IHOLD = 2, the triangular data

hold, which delivers an output that is concurrent with the input. Whenever possible, the triangular hold should be used (see Fig. 1).

These subjects are discussed further in FSE Program Summaries #5 and #6.

The transition equations for the three different data hold assumptions are given as follows. Here the temporal subscript "k" implies "t_k, the "current input point."

IHOLD = 0 (Zero Order Hold, Advancing)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k+1} = [\Gamma] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_k$$

Note that the output temporal subscript "k+1" is in advance of the input subscript "k".

IHOLD = 1 (First Order Hold, Advancing)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k+1} = [\Gamma] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_k + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} \frac{(u_k - u_{k-1})}{T}$$

Note that the output temporal subscript "k+1" is in advance of the most-recent input subscript "k".

IHOLD = 2 (Triangular Hold, Concurrent)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_k = [\Gamma] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}_{k-1} + \begin{bmatrix} \lambda_2 \\ \lambda_3 \\ \vdots \\ \lambda_{N+1} \end{bmatrix} u_{k-1} + \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_N \end{bmatrix} \frac{(u_k - u_{k-1})}{T}$$

Note that (only) in this formulation the output temporal subscript "k" is the same as the most-recent input subscript (concurrent).

Newton-Gregory Formulations

The above three data-hold formulations may be checked by comparisons with results obtained using the pertinent Newton-Gregory collating polynomials. These formulations are given by,

$$f_0(z) = Z\left\{\left(\frac{1 - e^{-sT}}{s}\right) f(s)\right\} \qquad \text{Z.O.H., Advancing}$$

$$f_1(z) = Z\left\{\left(\frac{1 - e^{-sT}}{s}\right)^2\left(\frac{1 + Ts}{T}\right) f(s)\right\} \qquad \text{F.O.H., Advancing}$$

$$f_2(z) = Z\left\{\left(\frac{1 - e^{-sT}}{s}\right)^2\left(\frac{e^{sT}}{T}\right) f(s)\right\} \qquad \text{T.O.H., Concurrent}$$

First Order Example

If the Laplace function is given by,

$$f(s) = \frac{1}{s + a}$$

the Newton-Gregory polynomials produce for the different data holds,

$$f_0(z) = \frac{1 - e^{-aT}}{a(z - e^{-aT})}$$

$$f_1(z) = \frac{aT + (aT - 1)(1 - e^{-aT}) + (1 - aT - e^{-aT})z^{-1}}{a^2T(z - e^{-aT})}$$

$$f_2(z) = \frac{aT - 1 + e^{-aT} + [1 - (1 + aT)e^{-aT}]z^{-1}}{a^2T(1 - e^{-aT}z^{-1})}$$

and substituting these expressions into the IHOLD formulations produces,

$$\gamma_{11} = e^{-aT}$$

$$\lambda_1 = (e^{-aT} + aT - 1)/a^2$$

$$\lambda_2 = (1 - e^{-aT})/a$$

regardless of the data hold (IHOLD is not input to XFRSET). Numerically, these quantities are delivered in the buffer SETB by a call to subroutine XFRSET (note that in this case $N^2 + N + 1 = 3$). XFRSET is a smart routine. As the product "aT" vanishes,

XFRSET produces the proper limits to these expressions,

$$\gamma_{11} \cong 1 - aT$$

$$\lambda_1 \cong \tfrac{1}{2}T^2$$

$$\lambda_2 \cong T(1 - \tfrac{1}{2}aT)$$

Hence, for f(s) = 1/s the zero, first, and triangular data hold formulations produce the Euler, Adams-Bashforth, and triangular integration algorithms, respectively. You should note that only in this single example does the Tustin formulation reduce to any of these state-space formulations (triangular hold).

Transfer Function Applicability

The quantity NTERM may be used as a check in determining the applicability of arbitrary transfer functions to real-time simulation in terms of a required value for "T" with respect to the coefficients a_n. This is discussed in ref. 2, where NTERM is identified as closely related to the "average discrete real-pole radius." The quantity is a measurement of both series convergence, and applicability. If NTERM is very large then it is certain that at least one pole in the transfer function is too large for the required cycle time.

An example is provided in Fig. 2 using a general second-order model. It should be noticed that the damping range (ζ) selected convolutes the formulation through five distinct z-transform solution regions. XFRSET has no problem with this. Bounded as well as unbounded functionalities are handled; the workload only increases as the average discrete real-pole radius increases. This is defined in ref. 1 as being the approximate function of the real poles,

$$\lambda = \Sigma \left| \mathrm{Re}(p_i) \right| T$$

for a wide class of functions, where "smaller poles tend to accelerate convergence." It is also noted that "the larger the order of the system, the more rapid the convergence." In Fig. 2 it is seen that NTERM in the region of about 15 should raise an alarm for second-order systems. In practice, a limit is placed on the size of λ in order to observe the system frequency content (considering the transition interval T used). A large value for NTERM informs you that convergence is slow. Either select a smaller value for T (sometimes possible using multirate techniques), or purge the transfer function of unrealistic poles.

Calls to XFRSET are computationally expensive. Hence, the workload should be appropriately distributed in a real-time simulation model. For example, if there are two transfer functions with different nonstationary denominators in a model, the calls to XFRSET should alternate. In this fashion the penalty of only one call accrues to any given real time cycle. This technique is possible, of course, because the coefficients should never be so dynamic that they require an XFRSET call every cycle.

The quantity NTERM (and hence the time penalty) is a function of various parameters. A very slow class of non-real-time computers has been used to help quantify this penalty, and demonstrate the futility of simulating unrealistic transfer functions. This is shown in Fig. 3, with parameters relevant to an IBM/AT class computer. For the second order system (N = 2) discussed above, we see that for typical systems ($0 \leq \zeta \leq 1$), where

Fig. 2 – XFRSET Convergence, Second–Order Model

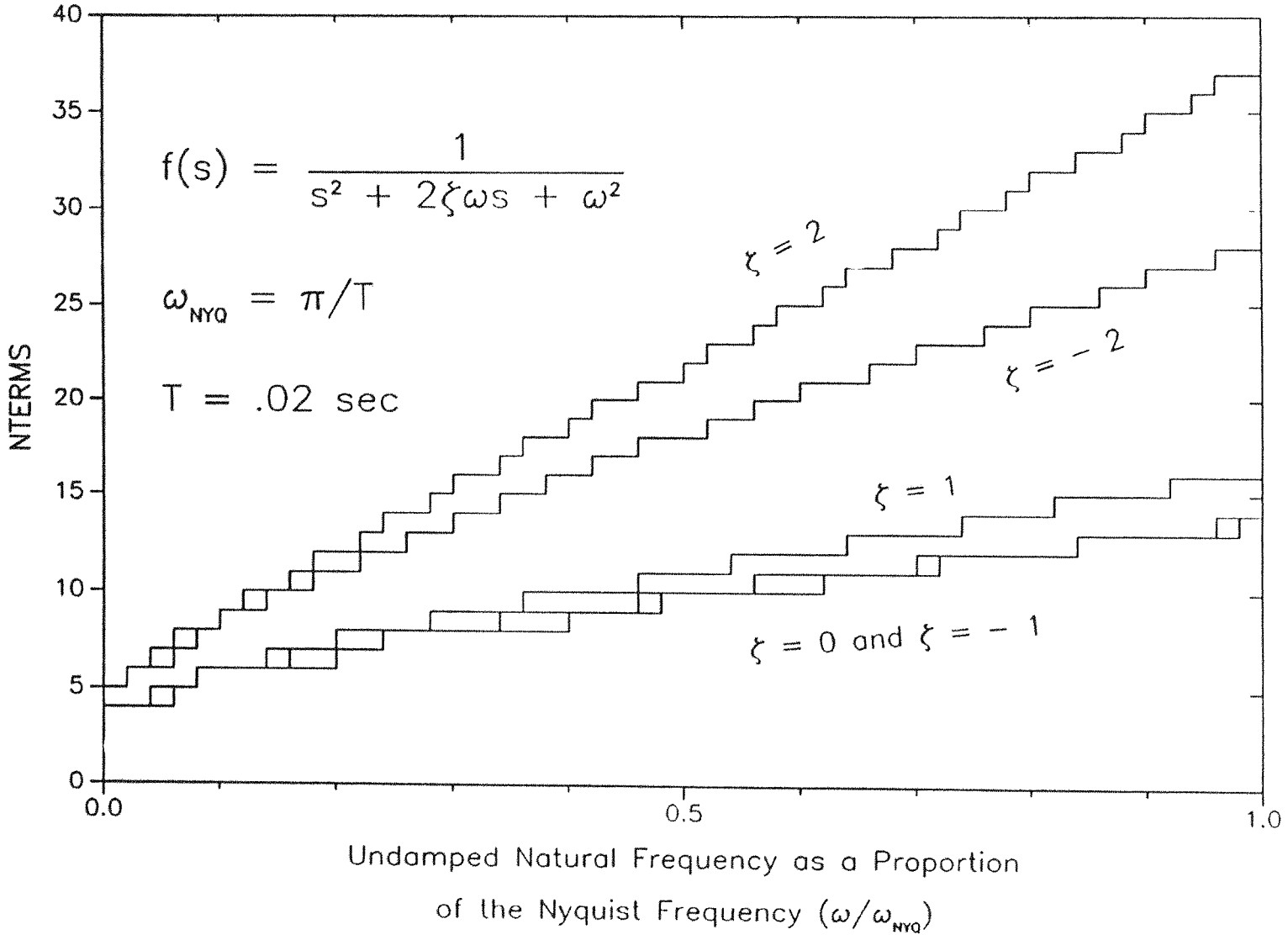
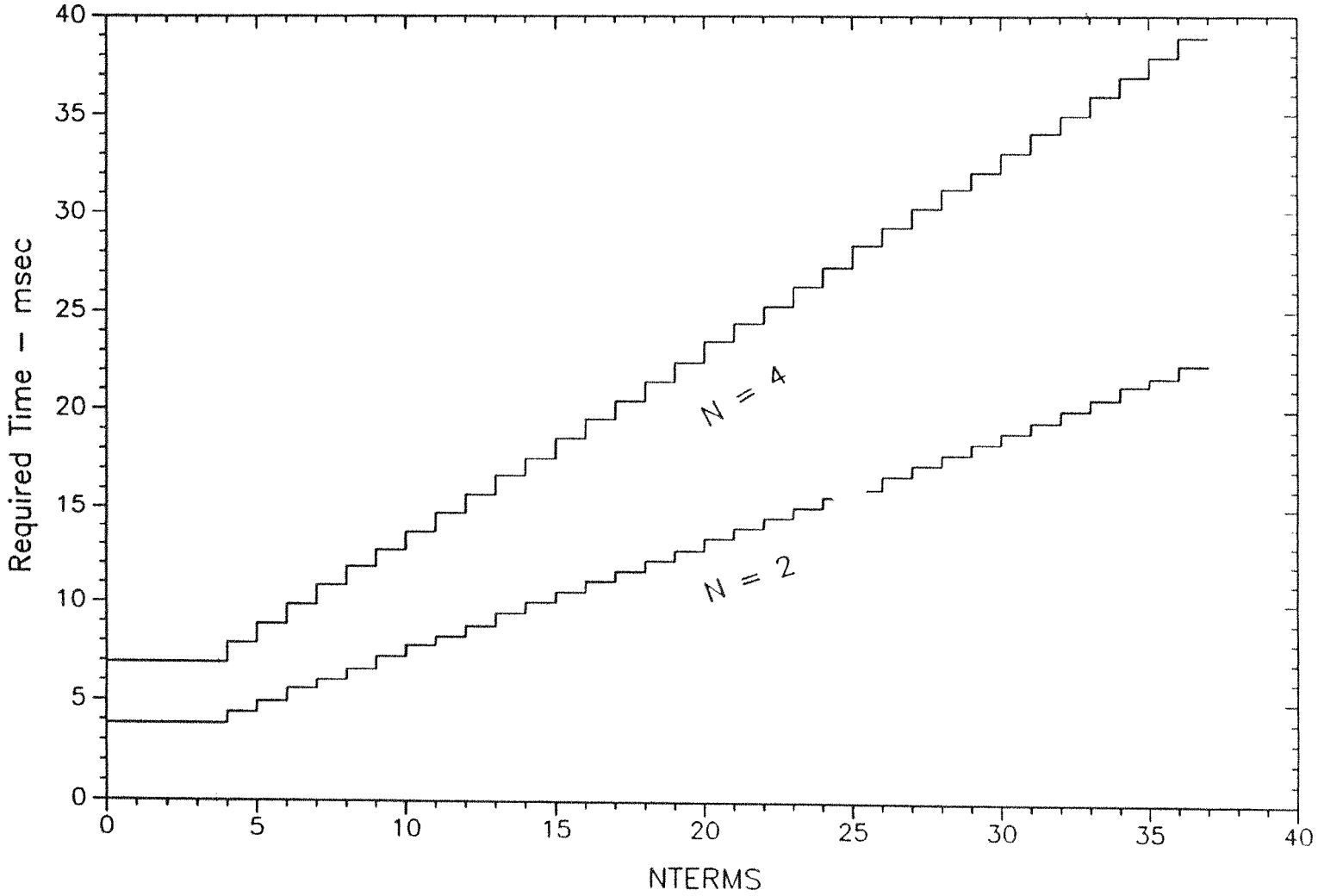


Fig. 3 – XFRSET Timing on an IBM/AT Computer



the undamped natural frequency is usually less than about 20% of the Nyquist frequency, an NTERM of less than about 7 is generally produced, and on an AT computer this would require about 6 msec. If the system were squared (N = 4), an AT computer would require about 10 msec. Note that the time penalty does not increase linearly with the order of the transfer function. Subroutine XFRSET *loves* high order systems.

Of course, our real-time computers are more than an order of magnitude faster than the speed of an AT computer. Hence, the data of Fig. 3 must include some scale factor if it is used to estimate timing. Also, the data of Fig. 2 includes some very unrealistic transfer functions, and values for NTERM on the order of 20 should never be anticipated.

Concluding Remarks

The program XFRSET produces the state transition matrix and the forcing function vector in a form that may be applied to *any order of data hold*. For real-time simulation it is best to restrict transfer functions to just those three data-hold formulations mentioned here.

For high-order transfer functions the determination of their z-transform equivalents involves a formidable amount of algebra. For this reason the XFRSET software is very valuable.

Whenever transfer functions can be combined, they should be combined, and a state space technique should be applied. Algebraic substitution techniques such as "Tustin" defeat this procedure and produce inferior results. If this latter bilinear technique is used in XFRRUN (i.e., IHOLD=3), then XFRSET should not be called; also, a state space solution is not produced.

For a constant coefficient case XFRSET need be only called once. For the slowly-varying coefficient case XFRSET should be called periodically.

Once created by a call to XFRSET, the output buffer SETB may be used by any number of transfer functions having the same denominator coefficients a_n.

Real-time transition is best handled by subroutine XFRRUN, outlined in FSE Program Summary #5. However, the user is invited to experiment with alternate transition schemes, using the XFRSET output buffer SETB.

FSE Program Summary #6 explains subroutine XFRBOD. This routine produces Bode plot data for transfer functions, as used in real-time simulation with the above techniques. Baseline (continuum) data is also generated.

```
C
      SUBROUTINE XFRSET(DT,N,A,NTERM,SETB)
C
C R.E.MCFARLAND -NASA-  JULY 18, 1989
C VERSION 1.0, OCT. 29, 1990
C 'STOP' REMOVED, REPLACED BY 'RETURN'.
C
C ROUTINE LIKE -FACT- EXCEPT DIFFERENT SETB DEFINITIONS,
C PERMITTING ZERO, FIRST OR TRIANGULAR HOLD TRANSITION...
C
C USES SUBROUTINE BTYPE FOR ITS ERROR MESSAGES.
C
C INPUTS:
C         DT      (CYCLE TIME IN SECONDS)
C         N       (ORDER OF THE DENOMINATOR POLYNOMIAL IN S)
C         A       (DIMENSION AT LEAST N)
C
C OUTPUTS:
C         NTERM   (A MEASURE OF CONVERGENCE SPEED)
C         SETB    (DIMENSION N**2 + N + 1 AT LEAST)
C
C XFRSET SETS UP THE TRANSITION MATRIX AND
C FORCING FUNCTION VECTOR WHICH WILL BE REQUIRED TO SOLVE FOR
C THE TRANSITION OF X TO U IN REAL TIME, AS IN:
C
C
C  ************************************
C  *                                  *
C  *  U                1              *
C  *  --- = ------------------------------------------  *
C  *  X      S**N + A(N)*S**(N-1) + ... + A(2)*S + A(1)      *
C  *                                  *
C  ************************************
C
C
C
      DIMENSION A(1),SETB(1)
C
C MAXIMUM INTERNAL DIMENSIONS.  GREATER DIMENSIONS MAY BE USED.
      DIMENSION HP(20),P(20),S(20),BB(20)
C
C DOUBLE PRECISION HAS BEEN MINIMIZED.
      DOUBLE PRECISION P,S,E,F,GSUM
C
C*********************************************************************
C*********************************************************************
C
C   FUNCTIONS ONLY OF THE DENOMINATOR POLYNOMIAL:
C
C BUFFER DEFINITIONS:      (SIZE = N**2 + N + 1)
C
```

```
C SETB(1)       C1    FIRST ELEMENT, FORCING FUNCTION VECTOR
C SETB(2)       C2    SECOND ELEMENT, FORCING FUNCTION VECTOR
C ...           ...   ...
C SETB(N+1)     CN+1  N + 1  ELEMENT, FORCING FUNCTION VECTOR
C
C SETB(N+2)     F(1,1)  TRANSITION MATRIX
C SETB(N+3)     F(1,2)  TRANSITION MATRIX
C ...           ...   ...
C SETB(I+1+J*N)  F(J,I)  TRANSITION MATRIX
C ...           ...   ...
C SETB(N**2+N+1)  F(N,N)  TRANSITION MATRIX
C
C
C SEE: "ON OPTIMIZING COMPUTATIONS FOR TRANSITION MATRICES"
C     IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. AC-23,
C     NO. 3, JUNE 1978.
C
C AND: "FSE PROGRAM SUMMARY DOCUMENT #4", XFFSET, FEB. 1990
C
C AND: "FSE PROGRAM SUMMARY DOCUMENT #5", XFFRUN, FEB. 1990
C
C
C NTERM - A MEASURE OF THE SPEED OF SERIES CONVERGENCE,
C       (AN INTERNAL VARIABLE).  SEE IEEE DOC IF INTERESTED.
C
C SIX DECIMAL DIGITS ACCURACY IN DETERMINING PARAMETERS.
    DATA PFM5/0.000005/
C
C TRANSFER FUNCTION RESTRICTION (ACCORDING TO DIMENSION STATEMENTS)
    IF(N.LE.0) RETURN
    IF(N.GT.20) THEN
    CALL BTYPE(40,' XFRSET: ARBITRARY LIMIT OF 20TH ORDER ')
    RETURN
    END IF
C N = 1 SPECIAL CASE (AND MATHEMATICAL LIMIT OF SINGLE INTEGRATION)
C SETB(3) = PHI(1,1)
    IF(N.GT.1) GO TO 500
    CCHECK = A(1)*DT
    IF(ABS(CCHECK).GT.1.0E-6) THEN
     SETB(3) = EXP(-CCHECK)
     SETB(2) = (1.0 - SETB(3))/A(1)
     SETB(1) = (DT - SETB(2))/A(1)
    ELSE
     SETB(3) = 1.0 - CCHECK
     SETB(2) = DT*(1.0 - 0.5*CCHECK)
     SETB(1) = DT**2*0.5
    END IF
    RETURN
 500 CONTINUE
C
C COMPUTE LEADING FACTORIAL TERMS FOR EACH ELEMENT OF
```

```
C FIRST COLUMN OF TRANSITION MATRIX.
    HL = 1.0
    XFACT = 1.0
    NP1 = N + 1
C
    DO 1000 I=1,N
     NP1MI    = NP1 - I
     FACTOR   = HL/XFACT
     SETB(I+NP1) = FACTOR
     HP(I)    = HL
     HL       = HL*DT
     XFACT    = XFACT*I
     BB(NP1MI) = - HL*A(NP1MI)
 1000 CONTINUE
C
C AT THIS POINT, FACTOR = DT**(N-1)/(N-1)! TO BE USED BELOW.
C
C OBTAIN KSTART.
     DO 1100 KSTART=1,N
     IF(A(KSTART).NE.0.0) GO TO 1200
 1100  CONTINUE
C
C AT THIS POINT ALL COEFFICIENTS ARE ZERO AND SERIES IS ALSO ZERO.
C HENCE, SPECIAL FORMULA FOR FORCING VECTOR.
    NTERM = 0
    KSTART = NP1
    GO TO 1800
C
C WILL FIND VALUE FOR NTERM BELOW.
 1200 CONTINUE
C
C INITIALIZE SERIES SUM.
    DO 1300 I=KSTART,N
     P(I) = BB(I)/XFACT
     S(I) = P(I)
 1300 CONTINUE
C
C SUM SERIES.  100 TERM LIMIT BASED UPON EITHER REAL POLES TIMES
C CYCLE TIME GREATER THAN ABOUT 30, OR FREQUENCY MANY TIMES THE
C NYQUIST FREQUENCY.
C
     DO 1500 NTERM = 1,100
     RATIO = N + NTERM
     E    = 0.0
     IGO = 0
      DO 1400 J = KSTART,N
      F   = (E + BB(J)*P(N))/RATIO
      E   = P(J)
      P(J) = F
      S(J) = S(J) + F
      IF(ABS(F).LE.PFM5*ABS(S(J))) GO TO 1400
```

```
      IGO = 1
 1400   CONTINUE
C
C CALCULATING PSI SUB J
     IF(IGO.EQ.0) GO TO 1600
 1500  CONTINUE
C
    CALL BTYPE(40,' XFRSET:  RETHINK YOUR TRANSFER FUNCTION')
    RETURN
C
C USING A CYCLE TIME OF 0.1, FOR EXAMPLE, THE TRANSFER FUNCTION
C 1/(0.1*S + 1) REQUIRES NTERM = 3
C
 1600  CONTINUE
C
C ADD IN LEADING TERM.
     DO 1700 I=KSTART,N
     S(I)    = S(I) * HP(I)
C FIRST COLUMN OF TRANSITION MATRIX
     INP1    = I + NP1
     SETB(INP1) = SETB(INP1) + S(I)
 1700  CONTINUE
C
C COMPUTE REMAINING ROWS OF STATE TRANSITION MATRIX.
C STORE FORCING VECTOR COMPONENTS CONTAINING ELEMENTS OF SAME.
 1800 CONTINUE
C
    IF(N.LT.2) GO TO 2000
     DO 1900 J = 2,N
     NJ         = N*J - N - 1
     NJP1       = NJ + N
     DUM        = SETB(NJP1 + 2)
     SETB(J + 1)   = DUM
     SETB(NJP1 + 3) = - A(1)*DUM
C
      DO 1900 I = 2,N
      SETB(NJP1 + I + 2) = SETB(NJ + I + 1) - A(I)*DUM
 1900  CONTINUE
C
 2000 CONTINUE
C
C FORCING VECTORS COMPUTED.
    IF(KSTART.LE.N) GO TO 2100
C
C ALL COEFFICIENTS ARE ZERO.  SPECIAL FORMULAS FOR FORCING VECTOR.
    SETB(2) = FACTOR*DT/N
    SETB(1) = SETB(2)*DT/NP1
    RETURN
C
C NOT ALL COEFFICIENTS ARE ZERO.  REGULAR FORMULAS.
 2100 CONTINUE
```

```
      APIVOT = - 1.0/A(KSTART)
      SETB(2) = APIVOT*S(KSTART)
C
C FORCING VECTOR
      IP  = KSTART - 1
      IPI = KSTART + 1
      IF(IPI.GT.N) GO TO 2200
      KSUM = IPI
      GSIG = S(IPI)
      GO TO 2300
C
 2200 CONTINUE
      KSUM   = N
      FACTOR = FACTOR*DT/KSTART
      GSIG   = SETB(NP1 - IP) - FACTOR
C
 2300 CONTINUE
      GSUM = 0.0
      IF(IPI.GT.KSUM) GO TO 2500
C
       DO 2400 K = IPI,KSUM
       GSUM = GSUM + A(K)*SETB(K - IP)
 2400   CONTINUE
C
 2500 CONTINUE
      SETB(1) = APIVOT*(GSIG + GSUM)
      RETURN
      END
```